Docket No. AUS919990655US1 ***PATENT***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Achtermann et al.**	§	
	§	Group Art Unit: **2157**
Serial No. **09/438,436**	§	
	§	Examiner: **Todd, Gregory G.**
Filed: **November 12, 1999**	§	
	§	
For: **Apparatus for Connection Management and the Method Therefor**	§	
	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on August 11, 2006.

No fees are believed to be necessary. If, however, any fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-4, 6-15, 17-26 and 28-33.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: 5, 16 and 27.
2. Claims withdrawn from consideration but not canceled: NONE.
3. Claims pending: 1-4, 6-15, 17-26 and 28-33.
4. Claims allowed: NONE.
5. Claims rejected: 1-4, 6-15, 17-26 and 28-33.
6. Claims objected to: NONE.

C. CLAIMS ON APPEAL

The claims on appeal are: 1-4, 6-15, 17-26 and 28-33.

STATUS OF AMENDMENTS

There are no amendments after the final rejection dated May 23, 2006.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1:

The invention embodiment claimed in Appellants' independent claim 1 provides a connection scheduling method (Appellants' specification, page 8, lines 3-12). The claimed connection scheduling method embodiment comprises determining if a job is available for scheduling (Appellants' specification, page 14, lines 1-14); determining if a session is available in response to said step of determining if said job is available (Appellants' specification, page 15, lines 1-3), wherein said session is included in a pool of sessions (Appellants' specification, page 15, lines 3-12), said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job (Appellants' specification, page 15, line 13 to page 16, line 2); launching said session to effect said execution of said job, if said session is available (Appellants' specification, page 19, lines 7-9 9); and launching an error handling thread in response to an error condition, said error handling thread releasing said session (Appellants' specification, page 20, lines 4-6).

Dependent claim 4:

The invention embodiment claimed in Appellants' independent claim 4 comprises determining if said connection is an existing connection (Appellants' specification, page 18, lines 4-6), and wherein said step of creating said connection is performed if said connection is not an existing connection (Appellants' specification, page 18, lines 14-15).

Dependent claim 6:

The invention embodiment claimed in Appellants' independent claim 6 comprises changing value of a job state from a first value to a second value in response to said launching of said error handling thread (Appellants' specification, page 20, lines 11-12).

Dependent claim 7:

The invention embodiment claimed in Appellants' independent claim 7 provides for said first value signals that said job is available for scheduling (Appellants' specification, page 17, line 13, to page 18, line 3).

Dependent claim 8:

The invention embodiment claimed in Appellants' independent claim 8 comprises retrying said steps of determining if a job is available for scheduling (Appellants' specification, page 19, lines 11-13), determining if a session is available (Appellants' specification, page 19, line 13), and launching said session, in response to an error condition (Appellants' specification, page 19, lines 13-16).

Dependent claim 9:

The invention embodiment claimed in Appellants' independent claim 9 comprises retrying is repeated until a predetermined time interval has elapsed (Appellants' specification, page 21, lines 6-8).

Independent claim 12:

The invention embodiment claimed in Appellants' independent claim 12 provides a data processing system for connection scheduling (Appellants' specification, page 8, lines 3-12). The claimed data processing system for connection scheduling provides circuitry operable for determining if a job is available for scheduling (Appellants' specification, page 14, lines 1-14) provides circuitry operable for determining if a session is available, in response to said circuitry operable for determining if said job is available (Appellants' specification, page 15, lines 1-3), wherein said session is included in a pool of sessions (Appellants' specification, page 15, lines 3-12), said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job (Appellants' specification, page 15, line 13 to page 16, line 2); provides circuitry operable for launching said session to effect said execution of said job, if said session is available (Appellants' specification, page 19, lines 7-9); and provides circuitry operable for launching an error handling thread in response to an error condition, said error handling thread releasing said session (Appellants' specification, page 20, lines 4-6).

The system recited in claim 12, as well as dependent claims 13-15 and 19-22, may be a bus system comprised of system bus **212**; I/O adapter **218**; communication adapter **234**, memory comprised of read only memory **216** and random access memory **214**, and central processing unit **210** performing the steps described in the specification at page 13, line 7, to page 23, line 3, or equivalent.

Dependent claim 15:

The invention embodiment claimed in Appellants' independent claim 15 provides circuitry operable for determining if said connection is an existing connection (Appellants' specification, page 18, lines 4-6), and wherein said circuitry operable for creating said connection is operated if said connection is not an existing connection (Appellants' specification, page 18, lines 14-15).

Dependent claim 17:

The invention embodiment claimed in Appellants' independent claim 17 provides for circuitry operable for changing value of a job state from a first value to a second value in response to said launching of said error handling thread (Appellants' specification, page 20, lines 11-12).

Dependent claim 18:

The invention embodiment claimed in Appellants' independent claim 18 provides for said first value signals that said job is available for scheduling (Appellants' specification, page 17, line 13, to page 18, line 3).

Dependent claim 19:

The invention embodiment claimed in Appellants' independent claim 19 provides circuitry operable for retrying said steps of determining if a job is available for scheduling (Appellants' specification, page 19, lines 11-13), determining if a session is available (Appellants' specification, page 19, line 13), and launching said session, in response to an error condition (Appellants' specification, page 19, lines 13-16).

Dependent claim 20:

The invention embodiment claimed in Appellants' independent claim 20 provides circuitry operable for retrying is operated until a predetermined time interval has elapsed (Appellants' specification, page 21, lines 6-8).

Independent claim 23:

The invention embodiment claimed in Appellants' independent claim 23 provides a computer program product embodied in a machine readable storage medium, the program product for job scheduling comprising instructions (Appellants' specification, page 8, lines 3-12). The claimed computer program product for job scheduling provides instructions for determining if a job is available for scheduling (Appellants' specification, page 14, lines 1-14); instructions for determining if a session is available in response to instructions for determining if said job is available (Appellants' specification, page 15, lines 1-3) , wherein said session is included in a pool of sessions (Appellants' specification, page 15, lines 3-12), said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job (Appellants' specification, page 15, line 13 to page 16, line 2); instructions for launching said session to effect said execution of said job, if said session is available (Appellants' specification, page 19, line 7-9); and instructions for launching an error handling thread in response to an error condition, said error handling thread releasing said session (Appellants' specification, page 20, lines 4-6).

A person having ordinary skill in the art would be able to derive computer instructions on a computer readable medium as recited in claim 23, as well as dependent claims 24-26 and 30-33, given **Figure 2** and the corresponding description at page 13, line 7, to page 23, line 3, without undue experimentation.

Dependent claim 26:

The invention embodiment claimed in Appellants' independent claim 26 provides instructions for determining if said connection is an existing connection (Appellants' specification, page 18, lines 4-6), and wherein said instructions for creating said connection are performed if said connection is not an existing connection (Appellants' specification, page 18, lines 14-15).

Dependent claim 28:

The invention embodiment claimed in Appellants' independent claim 28 provides instructions for changing value of a job state from a first value to a second value in response to said launching of said error handling thread (Appellants' specification, page 20, lines 11-12).

Dependent claim 29:

The invention embodiment claimed in Appellants' independent claim 29 provides for said first value signals that said job is available for scheduling (Appellants' specification, page 17, line 13, to page 18, line 3).

Dependent claim 30:

The invention embodiment claimed in Appellants' independent claim 30 provides instructions for retrying said steps of determining if a job is available for scheduling (Appellants' specification, page 19, lines 11-13), determining if a session is available (Appellants' specification, page 19, line 13), and launching said session, in response to an error condition (Appellants' specification, page 19, lines 13-16).

Dependent claim 31:

The invention embodiment claimed in Appellants' independent claim 31 provides instructions for retrying are repeated until a predetermined time interval has elapsed (Appellants' specification, page 21, lines 6-8).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION (Claims 1-3, 12-14, and 23-25)

Whether claims 1-3, 12-14, and 23-25 are unpatentable over Zolnowsky (U.S. Patent No. 6,779,182 B1) in view of Dorfman et al. (U.S. Patent No. 6,134,313) under 35 U.S.C. § 103(a).

B. GROUND OF REJECTION (Claims 4, 6-9, 15, 17-20, 26, and 28-31)

Whether claims 4, 6-9, 15, 17-20, 26, and 28-31 are unpatentable over Zolnowsky (U.S. Patent No. 6,779,182 B1) in view of Dorfman et al. (U.S. Patent No. 6,134,313) and further in view of Northrup (U.S. Patent No. 6,671,713 B2) under 35 U.S.C. § 103(a).

C. GROUND OF REJECTION (Claims 10-11, 21-22, and 32-33)

Whether claims 10, 11, 21, 22, 32, and 33 stand rejected under 35 U.S.C. § 103 as obvious over Zolnowsky (U.S. Patent No. 6,779,182 B1) in view of Dorfman et al. (U.S. Patent No. 6,134,313) and Northrup (U.S. Patent No. 6,671,713 B2) and further in view of Rangarajan et al. (U.S. Patent No. 6,260,077 B1).

ARGUMENT

A. GROUND OF REJECTION (Claims 1-3, 12-14, and 23-25)

Claim 1 is representative of the claims in this group and reads as follows:

1. A connection scheduling method comprising the steps of:
determining if a job is available for scheduling;
determining, in response to said step of determining if said job is available, if a session is available, wherein said session is included in a pool of sessions, said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job;
launching said session to effect said execution of said job, if said session is available; and
launching an error handling thread in response to an error condition, said error handling thread releasing said session.

Appellants respectfully submit that Zolnowsky and Dorfman, taken alone or in combination, fail to teach or suggest launching an error handling thread in response to an error condition, said error handling thread releasing said session in claim 1.

Zolnowsky is directed to a process scheduler or dispatcher for a multiprocessor system for real time applications. The Zolnowsky system uses a dispatcher model that maintains a dispatch queue for each processor and a separate global dispatch queue for unbound higher priority real time threads. Each processor has its own queue and a dispatcher. Each queue has a separate schedule lock associated with it to protect scheduling operations. A processor's dispatcher selects a thread for execution from one of the queues in the system as a candidate thread to execute. When a candidate thread is selected for execution, the processor proceeds to verify against threads in the global real time queue and the processor's own dispatch queue to select a highest priority runnable thread in the system.

Thus, the Zolnowsky system allows the dispatcher to prevent race conditions and minimize lock contention while assuring that high-priority threads are dispatched as quickly as possible. The Examiner acknowledges that Zolnowsky fails to teach launching an error handling thread in response to an error condition, said error handling thread releasing said session. However, the Examiner alleges that "the use and advantages for using such an error handling thread is well known to one skilled in the art at the time the invention was made as evidenced by the teachings of Dorfman" and that the presently claimed feature is taught by Dorfman in the following section:

> Referring back to FIG. 4, after the global objects are created at step 306, a main session is created at Step 308 to provide a clean runtime environment for a main application, such as an initialization application, which is loaded and posted to the main session at Step 310. A "session" binds together an execution context and system resources. The components of a preferred embodiment of a session 132 are illustrated in FIG. 11. The execution context includes an execution thread 200 on the processor 52 and a heap 206 which is a subset of the memory 54. In addition, the session 132 includes a session queue 202, an application, such as an idle application 208, and a Tcl interpreter 204 to be used by the applications running on the session 132.
>
> When the session 132 is created, the session queue 202 begins running. The session queue 202 can be used by other sessions to send messages into session 132. Initially, the session queue 202 waits for a message to arrive with an application to be executed on the session 132, and if one does not arrive within a predetermined time-out period, the session 132 exits. In the preferred embodiment, the main application locates the session application from the configuration manager 110 and sends this application to the session queue 202. The application is copied onto the heap 206 and will then be executed on the session 132. The application may utilize any resources allocated to the session 132, such as telephone lines, voice processing, fax processing, etc. The application will likely generate other events that will be posted to the session queue 202 to keep the session running. A user agent will usually be used by an application to calculate the next application to be executed in response to a given event. In addition, the session 132 provides exception handling for all applications that run on it. **When an exception occurs, the session 132 shuts down the execution thread 200 and releases all resources allocated to the session 132**.
>
> In the preferred embodiment, the resources allocated to the session 132 will be released automatically whenever the session 132 terminates, e.g., through normal termination or if the application executing on the session 132 crashes. In addition, the session 132 can start another session (i.e., a child session), and may "delegate" the use of some or all of its resources to this second session. The second session may utilize the delegated resources as if the resources had been allocated to the second session. However, in the preferred embodiment, the second session can release resources allocated to the second session, but cannot release delegated resources (i.e., those resources allocated to other sessions). When the second session terminates, the session 132 will regain control of the resources it delegated to the second session. **(emphasis added)**

(Dorfman, Column 11, *line 38, to column 12, line 20*)

Dorfman is directed to software architecture for a computer telephony server for simultaneously implementing a plurality of messaging applications is provided. The server further includes a session having a thread of execution on the processor and controlling a subset of the memory and a telephone line resource. The cited section of Dorfman describes the different components that are started for the execution of a main session and how an application is processed for running on the session. Dorfman also describes providing exception handling for the applications that run on the session and that when an exception occurs, **the session** shuts down the execution thread and releases all resources allocated to the session. The session, which Dorfman refers to as session **132**, is a main session that is created at Dorfman's step **308** to provide a clean runtime environment for a main application. Dorfman does not instantiate any additional threads. Thus, contrary to the Examiner's allegation that "the use and advantages for using such an error handling thread is well known to one skilled in the art at the time the invention was made as evidenced by the teachings of Dorfman" Dorfman does not launch an error handling thread, rather, Dorfman merely **shuts down** the execution thread. Thus, Zolnowsky and Dorfman, taken alone or in combination fail to teach or suggest launching an error handling thread in response to an error condition, said error handling thread releasing said session.

In response to these arguments, the Examiner states:

> In response to a) Zolnowsky does not teach the limitation. However, Dorfman clearly states that when an exception occurs, another thread is 'launched' to shut down the execution thread and further release all resources allocated to the session, and thus releasing the session (at least col. 11 line 38 – col. 12 line 20). Therefore, as can be seen, the process of shutting down the execution thread is in itself, a separate process of thread, said thread being "launched" when an exception or error does occur, otherwise, when there is no exception, the session continues without launching the 'exception thread' to shut down the execution thread.

Appellants respectfully submit that Dorfman does not teach or suggest that "another thread is 'launched' to shut down the execution thread and further release all resources allocated to the session, and thus releasing the session" as alleged by the Examiner. In fact the term launch does not appear in the cited section whatsoever. As discussed above, Dorfman teaches when an exception occurs, **the session** shuts down the execution thread and releases all resources allocated to the session.

Furthermore, no suggestion or incentive is present in either reference to modify the references to include such features. That is, there is no teaching or suggestion in Zolnowsky and

Dorfman, either alone or in combination, that a problem exists for which launching an error handling thread in response to an error condition, said error handling thread releasing said session, is a solution. To the contrary, Zolnowsky teaches a verification of a best possible thread selection by reconfirming the selected threads priority from the other threads in queue. Dorfman teaches shutting down the execution thread to releases all resources allocated to the session. Zolnowsky and Dorfman, either alone or in combination, do not recognize a need to perform the features, or similar features, as recited in claims 1, 12, and 23.

Moreover, neither reference teaches or suggests the desirability of incorporating the subject matter of the other reference. That is, no motivation is offered in either reference for the alleged combination. The Examiner alleges that the motivation for the combination is "this would enhance and allow the system of Zolnowsky to be prepared for error handling and as Dorfman teaches, when an exception does occur, it is desirable to release all running resources for the session." The Examiner acknowledges that Zolnowsky does not teach launching an error handling thread in response to an error condition, said error handling thread releasing said session. Dorfman merely shuts down the execution thread to release all the resources allocated to the session. Neither reference launches an error handling thread in response to an error condition. Thus, the only teaching or suggestion to even attempt the alleged combination is based on a prior knowledge of Appellants' claimed invention thereby constituting impermissible hindsight reconstruction using Appellants own disclosure as a guide.

One of ordinary skill in the art, being presented only with Zolnowsky and Dorfman, and without having a prior knowledge of Appellants' claimed invention, would not have found it obvious to combine and modify Zolnowsky and Dorfman to arrive at Appellants' claimed invention. To the contrary, even if one were somehow motivated to combine Zolnowsky and Dorfman, and it were somehow possible to combine the two systems, the result would not be the invention, as recited in claims 1, 12, and 23. The result shutting down the execution thread in response to an error that releases all the resources allocated to the session.

Thus, Zolnowsky and Dorfman, taken alone or in combination, fail to teach or suggest all of the features in independent claims 1, 12, and 23. At least by virtue of their dependency on claims 1, 11, and 19, the specific features of claims 2, 3, 13, 14, 24, and 25 are not taught or suggested by Zolnowsky and Dorfman, either alone or in combination. Accordingly, Appellants respectively request withdrawal of the rejection of claims 1-3, 12-14, and 23-25 under 35 U.S.C. § 103(a).

B. GROUND OF REJECTION (Claims 4, 6-9, 15, 17-20, 26, and 28-31)

B.1. Claims 4, 15, and 26

With regard to claims 4, 15, and 26 in this group of claims, the Examiner acknowledges that Zolnowsky and Dorfman do not explicitly disclose determining if said connection is an existing connection, and wherein said step of creating said connection is performed if said connection is not an existing connection. However, the Examiner alleges that Northrup teaches this feature in the following section:

> The communication primitives are built using the underlying computer operating system intraprocess and interprocess communication facilities and thus are operating-system-specific. On one operating system there may be, for example, five communication primitives supported, while another operating system may support twenty. A communication primitive generally must provide for several operations to be applied such as:
>
> Create: The ability to create an instance of the primitive
> Destroy: The ability to destroy an instance of the primitive
> Send: The ability to send data to the primitive
> Receive: The ability to receive data from the primitive
> Cycle: Send a default and receive default messages to/from the primitive
> Connect: Primitive specific connection function
> Disconnect: Primitive specific disconnection function
> Suspend: Primitive specific suspension function
> Resume: Primitive specific resumption function
>
> Communication primitives are registered with the Thread Communication Service for the specific operating system the TCS is executing on. The name, the location, and certain characteristics describing the communication primitive are retained by the TCS for subsequent use. In this context, the communication primitives become a reusable asset, needing to be developed and tested only one time.
>
> Each communication primitive has a shared object, referred to as the communication primitive object, describing the location of the various operations to be applied when using this primitive type. All primitives have the same communication primitive object structure. The TCS will load the communication primitive object at runtime only when requested for use by a communication point.

In a sense, the communication primitive can be thought of as analogous to the physical connection of a telephone on a phone network. A twisted pair

telephone would use one primitive while a cellular telephone would use a different primitive.

(Northrup, Column 4, lines 22-60)

The Examiner's cited section of Northrup teaches communication primitives that are the low-level mechanisms used to provide the physical communication between various processes. The processes participating in the communication are referred to as communication points. Two or more communication points are connected by a communication link using the communication primitives. There is nothing in this section, or any other section of Northrup, that teaches or suggests determining if a connection is an existing connection, and, if the connection is not an existing connection, creating a connection.

In view of the above, Zolnowsky, Dorfman, and Northrup, either alone or in combination, fail to teach or suggest the specific features recited independent claims 4, 15, and 26, or independent claims 1, 12, and 23, from which claims 4, 15, and 26 depend. Accordingly, Appellants respectfully request that the rejection of claims 4, 15, and 26 under 35 U.S.C. § 103 not be sustained.

B.2. __Claims 6, 17, and 28__

With regard to claims 6, 17, and 28 in this group of claims, the Examiner acknowledges that Zolnowsky and Dorfman do not explicitly disclose changing value of a job state from a first value to a second value in response to said launching of said error handling thread. However, the Examiner alleges that Northrup teaches this feature in the following sections:

Finally, the State Machine Thread Manager provides a special "Error" state to which a datum will transition when the exiting value of the State Thread is undefined in the current state machine.

(Northrup, Column 56, lines 33-36)

In this section, Northrup describes providing an error state to which a datum will transition when the exiting value of the state thread is undefined in the current state machine. However, the state change is not in response to the launching of an error handling thread.

> Using a standard Application Programming Interface, the Application Process can read the standard output of the logically named NEE to which it is ATTACHED. This is accomplished by the Minor Service Reader/Writer Thread of the NEEM that is reading the actual standard output of the NEE. It will send the standard output to the Application Process. Likewise, the Minor Service Reader/Writer Thread of the NEEM which reads the actual standard error will send the standard error output to the Application Process.

(Northrup, Column 55, lines 27-35)

In this section, Northrup describes using a standard Application Programming Interface (API) to read the output of the logically Named Execution Environment to which the API is attached, which may include reading a standard error. All this section teaches is reading an error and does not teach or suggest the launching of an error handling thread.

> When a new Service Thread is made available to the computer system, it can register its service by calling the Thread Directory Service specifying a REGISTER operation and providing the required information along with any optional information or attributes. Alternatively, a separate application can register other Service Threads available to the computer system by calling the Thread Directory Service and specifying a REGISTER operation along with the appropriate information. This permits a separate application program to provide this information without requiring the Service Thread to register itself. Duplicate entries in a given Thread Service Directory are not permitted. In this instance, the Thread Directory Service will return an error indication to the registering thread. In registering the Service Thread, the Thread Directory Service will assign a unique Thread Communication Identifier to be associated with the Service Thread. The Thread Directory Service will then return this identifier to the thread registering the service.

(Northrup, Column 27, line 66 to column 28, line 15)

In this section, Northrup describes the addition of new service threads. None of these sections teaches or suggests changing value of a job state from a first value to a second value **in response to said launching of said error handling thread**.

In view of the above, Appellants respectfully submit that Zolnowsky, Dorfman, and Northrup, neither alone nor in combination, teach or suggest the specific features recited independent claims 6, 17, and 28, or independent claims 1, 12, and 23, from which claims 6, 17, and 28 depend. Accordingly, Appellants respectfully request that the rejection of claims 6, 17, and 28 under 35 U.S.C. § 103 not be sustained.

B.3. <u>Claims 7, 18, and 29</u>

With regard to claims 7, 18, and 29 in this group of claims, the Examiner alleges that Zolnowsky teaches said first value signals that said job is available for scheduling at column 8, lines 11-17, shown above. As discussed above, this section of Zolnowsky describes the examination of priorities within the various queues and the use of a priority variable to indicate a maximum priority level and making use of a suitable synchronization algorithm to prevent miscommunication between a dispatch queue and a real time queue. However, Zolnowsky further teaches that any errors between the queues are caught in a verification step and that the verification of a best possible thread selection by reconfirming the selected threads priority from the other threads in queue. There is nothing in these sections, or any other section of Zolnowsky, that teaches a first value that signals that a job is available for scheduling.

In view of the above, Zolnowsky, Dorfman, and Northrup, either alone or in combination, fail to teach or suggest the specific features recited independent claims 7, 18, and 29, or independent claims 1, 12, and 23, from which claims 7, 18, and 29 depend. Accordingly, Appellants respectfully request that the rejection of claims 7, 18, and 29 under 35 U.S.C. § 103 not be sustained.

B.4. <u>Claims 8, 19, and 30</u>

With regard to claims 8, 19, and 30 in this group of claims, the Examiner alleges that Zolnowsky teaches retrying said steps of determining if a job is available for scheduling, determining if a session is available, and launching said session, in response to an error condition at column 8, lines 11-17, shown above. As discussed above, this section of Zolnowsky describes the verification of the priorities of threads and, if a higher priority thread exist, substituting the higher priority thread. The Examiner further alleges that Dorfman teaches starting another session in response to an error condition. As discussed above, Dorfman teach shutting down the execution thread to release the resources allocated to the session.

In view of the above, Zolnowsky, Dorfman, and Northrup, either alone or in combination, fail to teach or suggest the specific features recited independent claims 8, 19, and 30, or independent claims 1, 12, and 23, from which claims 8, 19, and 30 depend. Accordingly, Appellants respectfully request that the rejection of claims 8, 19, and 30 under 35 U.S.C. § 103 not be sustained.

B.5. Claims 9, 20, and 31

With regard to claims 9, 20, and 31 in this group of claims, the Examiner acknowledges that Zolnowsky and Dorfman do not explicitly disclose retrying is repeated until a predetermined time interval has elapsed. However, the Examiner alleges that the use and advantages for retrying tasks based upon elapsed time is well known to one skilled in the art at the time the invention was made as evidence by the teachings of Northrup in the following section:

> The Application Process uses the Configuration Administrator Minor Service to administer zero or more components of software from shared libraries. Each component is said to offer a Minor Service. The specifications for the administration of the Minor Services can be provided directly by the Application Service, or, indirectly through a data store monitored by the Configuration Administrator. These specifications can instruct the Configuration Administrator Minor Service to perform the desired operation immediately, at a predefined time (which may be an interval), or, as a result of some event which is later communicated to the Configuration Administrator Minor Service.
>
> The Configuration Administrator Minor Service provides the following operations:
>
> 1. Locates specified Minor Services
> 2. Loads specified Minor Services
> 3. Executes specified Minor Services
> 4. Establishes communication channel with the specified Minor Service.
> 5. Suspends execution of specified Minor Services
> 6. Resumes execution of specified Minor Services
> 7. Replaces specified Minor Service with a new Minor Service rerouting communication channels as appropriate
> 8. Unloads specified Minor Service
> 9. Provides for manual state retention between replaceable Minor Services
> 10. Notification
>
> Note that the Configuration Administrator Minor Service operations can be specified to occur at set time intervals; at predefined time periods; as a result of external events; or, as a result of internal events. Events, in this context are

registered with the Configuration Administrator Minor Service to denote their occurrence.

(Northrup, Column 10, line 49 to column 11, line 18)

While this section of Northrup may teach performing desired operations at a predefined time, this section does not teach or suggest repeating retrying the steps of determining if a job is available for scheduling, determining if a session is available, and launching a session, **in response to an error condition**, until a predetermined time interval has elapsed. Appellants respectfully submit that it would not be obvious to one skilled in the art at the time the invention was made to retry tasks based upon elapsed time in response to an error condition, as neither of the applied references teach or suggest this feature. Thus, Zolnowsky, Dorfman, and Northrup, either alone or in combination, fail to teach the features recited in claims 9, 20, and 31.

In view of the above, Zolnowsky, Dorfman, and Northrup, either alone or in combination, fail to teach or suggest the specific features recited independent claims 9, 20, and 31, or independent claims 1, 12, and 23, from which claims 9, 20, and 31 depend. Accordingly, Appellants respectfully request that the rejection of claims 9, 20, and 31 under 35 U.S.C. § 103 not be sustained.

C. GROUND OF REJECTION (Claims 10, 11, 21, 22, 32, and 33)

Claims 10, 11, 21, 22, 32 and 33 are dependent on independent claims 1, 12, and 23 and, thus, these claims distinguish over Zolnowsky and Dorfman for at least the reasons noted above with regards to claims 1, 12, and 23. Northrup and Rangarajan are not cited by the Examiner as teaching or suggesting the launching an error handling thread in response to an error condition, said error handling thread releasing said session. Accordingly, claims 10, 11, 21, 22, 32, and 33 are patentable for the reasons state previously with regard to claims 1, 12, and 23.

CONCLUSION

In view of the above, Appellants respectfully submit that claims 1-4, 6-15, 17-26 and 28-33 are allowable over the cited prior art and that the application is in condition for allowance. Accordingly, Appellants respectfully request the Board of Patent Appeals and Interferences to reverse the rejections set forth in the Final Office Action.

/Francis Lammes/
Francis Lammes
Reg. No. 55,353
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A connection scheduling method comprising the steps of:

determining if a job is available for scheduling;

determining, in response to said step of determining if said job is available, if a session is available, wherein said session is included in a pool of sessions, said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job;

launching said session to effect said execution of said job, if said session is available; and

launching an error handling thread in response to an error condition, said error handling thread releasing said session.

2. The method of claim 1 wherein said session comprises a thread.

3. The method of claim 1 further comprising the step of creating a connection to a target system for said execution of said job.

4. The method of claim 3 further comprising the step of determining if said connection is an existing connection, and wherein said step of creating said connection is performed if said connection is not an existing connection.

6. The method of claim 1 further comprising the step of changing value of a job state from a first value to a second value in response to said launching of said error handling thread.

7. The method of claim 6 wherein said first value signals that said job is available for scheduling.

8. The method of claim 1 further comprising the step of retrying said steps of determining if a job is available for scheduling, determining if a session is available, and launching said session, in response to an error condition.

9. The method of claim 8 wherein said step of retrying is repeated until a predetermined time interval has elapsed.

10. The method of claim 9 further comprising the step of registering a callback method in response to an expiry of said predetermined time interval.

11. The method of claim 10 wherein said steps of determining if a job is available for scheduling, determining if a session is available, and launching said session are performed in response to an invoking of said callback method by a target system, said target system for execution of said job.

12. A data processing system for connection scheduling comprising:

circuitry operable for determining if a job is available for scheduling;

circuitry operable for determining, in response to said circuitry operable for determining if said job is available, if a session is available, wherein said session is included in a pool of sessions, said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job;

circuitry operable for launching said session to effect said execution of said job, if said session is available; and

circuitry operable for launching an error handling thread in response to an error condition, said error handling thread releasing said session.

13. The system of claim 12 wherein said session comprises a thread.

14. The system of claim 12 further comprising circuitry operable for creating a connection to a target system for said execution of said job.

15. The system of claim 14 further comprising circuitry operable for determining if said connection is an existing connection, and wherein said circuitry operable for creating said connection is operated if said connection is not an existing connection.

17. The system of claim 12 further comprising circuitry operable for changing value of a job state from a first value to a second value in response to said launching of said error handling thread.

18. The system of claim 17 wherein said first value signals that said job is available for scheduling.

19. The system of claim 12 further comprising circuitry operable for retrying said steps of determining if a job is available for scheduling, determining if a session is available, and launching said session, in response to an error condition.

20. The system of claim 19 wherein said circuitry operable for retrying is operated until a predetermined time interval has elapsed.

21. The system of claim 20 further comprising circuitry operable for registering a callback method in response to an expiry of said predetermined time interval.

22. The system of claim 21 wherein said circuitry operable for determining if a job is available for scheduling, determining if a session is available, and launching said session are operated in response to an invoking of said callback method by a target system, said target system for execution of said job.

23. A computer program product embodied in a machine readable storage medium, the program product for job scheduling comprising instructions for:

determining if a job is available for scheduling;

determining, in response to instructions for determining if said job is available, if a session is available, wherein said session is included in a pool of sessions, said pool of sessions having a preselected one of a set of priority levels corresponding to a priority level of said job and wherein said session effects an execution of said job;

launching said session to effect said execution of said job, if said session is available; and

launching an error handling thread in response to an error condition, said error handling thread releasing said session.

24. The program product of claim 23 wherein said session comprises a thread.

25. The program product of claim 23 further comprising instructions for creating a connection to a target system for said execution of said job.

26. The program product of claim 25 further comprising instructions for determining if said connection is an existing connection, and wherein said instructions for creating said connection are performed if said connection is not an existing connection.

28. The program product of claim 23 further comprising instructions for changing value of a job state from a first value to a second value in response to said launching of said error handling thread.

29. The program product of claim 28 wherein said first value signals that said job is available for scheduling.

30. The program product of claim 29 further comprising programming for retrying said steps of determining if a job is available for scheduling, determining if a session is available, and launching said session, in response to an error condition.

31. The program product of claim 30 wherein said instructions for retrying are repeated until a predetermined time interval has elapsed.

32. The program product of claim 31 further comprising programming for registering a callback method in response to an expiry of said predetermined time interval.

33. The program product of claim 32 wherein said instructions for determining if a job is available for scheduling, determining if a session is available, and launching said session are executed in response to an invoking of said callback method by a target system, said target system for execution said job.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.